# A Nonlinear Programming Algorithm for Optimizing Conventional Beamformer Shading Weights

Thomas A. Wettergren
John P. Casey
NUWC Division Newport

Charles M. Traweek
Office of Naval Research

NAVSEA
NEWPORT
Undersea Warfare Center Division
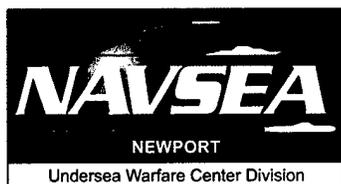
Naval Undersea Warfare Center Division
# Newport, Rhode Island

20040113 099

PREFACE

This report was prepared under Project No. W280033, "EA89 TW Support," principal investigator Thomas A. Wettergren (Code 2002). The sponsoring activity is the Office of Naval Research (ONR 321SS, C. M. Traweek).

The technical reviewer for this report was Deepak V. Ramani (Code 2133).

Reviewed and Approved: 10 November 2003
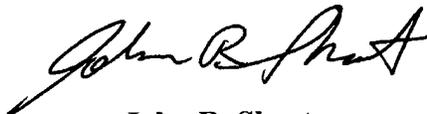
John R. Short
Director, Submarine Combat Systems

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 10 November 2003	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE

A Nonlinear Programming Algorithm for Optimizing Conventional Beamformer Shading Weights

5. FUNDING NUMBERS

6. AUTHOR(S)
Thomas A. Wettergren
John P. Casey
Charles M. Traweek

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Undersea Warfare Center Division
1176 Howell Street
Newport, RI 02841-1708

8. PERFORMING ORGANIZATION REPORT NUMBER

TR 11,466

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
Ballston Centre Tower One
800 No. Quincy Street
Arlington, VA 22217-5660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A numerical method to determine the optimal shading weights for a conventional delay-and-sum beamformer has been developed. The method employs a maximization of the deflection coefficient under the constraint of maintaining signal gain. This provides an optimal array shading scheme based on available noise data. The algorithm has been implemented using off-the-shelf numerical methods that are applicable only for small arrays. For large arrays, a special-purpose optimization algorithm has been developed. The performance of the algorithms on measured test array data is included to show the level of performance improvement.

14. SUBJECT TERMS

Sonar Arrays   Passive Sonar Systems   Beamformer Shading Weights
Numerical Optimization   Nonlinear Programming Problems

15. NUMBER OF PAGES
30

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

A NONLINEAR PROGRAMMING ALGORITHM FOR OPTIMIZING CONVENTIONAL BEAMFORMER SHADING WEIGHTS

1. INTRODUCTION

Environmental adaptation of sonar systems is the process of improving the performance of a sonar system given some known environmental conditions (such as measured environmental noise). Since a well-designed system is usually environmental-noise limited, the use of this knowledge of the environment has the potential for large performance improvements. An effort has been undertaken to develop an optimization system to improve the performance of conventional beamformers under known environmental noise conditions. The resulting optimization algorithm shows great potential for improving passive sonar system performance against unknown signals when the (element-level) noise performance is known. This algorithm is directly applicable to the conventional delay-and-sum beamformer and thus requires minimal system impact for implementation.

The use of delay-and-sum beamformers is commonplace in passive sonar systems design. These are often referred to as conventional beamformers. The conventional beamformer functions by applying a time delay to each sensor and multiplying each of these by a prescribed weight before summing the element responses to provide the beamformer output. Traditionally, the beamformer is designed so that the time delays maximize the sensitivity of the array to a desired (look) direction, thus achieving robust signal gain; and the corresponding weightings are chosen to provide low sensitivity to energy that does not match with the incident wave (low sidelobes), which achieves low noise response.

In the absence of specific knowledge of the noise field encountered, the conventional beamformer design strategy provides robust performance. However, components of the noise field are often predictable, particularly in the case of arrays that employ long-duration time averaging. Those arrays have the opportunity to observe the noise field (as seen by the array elements) in real time. In this case, adaptive beamforming techniques are often employed, wherein the beamformer converts each of the received sensor outputs into a digital form and employs frequency-domain calculations to cancel those components of the received signal that appear to be attributable to noise. Such calculations provide a dramatic change of the structure of the beamformer to provide this noise removal. As an alternative means of reducing the impact of noise on the beamformer, environmentally adaptive methods of conventional beamforming are available, where the conventional beamformer structure is maintained, and the shading weights found within that structure are optimized for best performance against the noise field.

2. OBJECTIVE FUNCTION FOR CONVENTIONAL SHADING WEIGHT OPTIMIZATION

The problem of conventional beamformer shading weight optimization has been recently formulated from a deflection optimization perspective (see reference 1). Deflection is a measurement of the statistical variation between two hypotheses in a binary (two-state) statistical hypothesis test. In the case of passive arrays, these states are the state where signal is present and the state where signal is absent. The deflection coefficient is a parameter that provides a measurement of this variation. For the case of distinguishing signal-plus-noise (hypothesis 1) from noise alone (hypothesis 0), one assumes that both processes are governed by Gaussian random variables with the same variance. Such an assumption is reasonable in the case of small signal-to-noise ratio (SNR) that is seen in passive sonar system applications. In that case, the deflection coefficient is given by

$$d = \frac{\mu_{SN} - \mu_N}{\sigma_N},$$

(1)

where μ_{SN} is the mean under the signal-plus-noise hypothesis, μ_N is the mean under the noise-only hypothesis, and σ_N is the standard deviation of the noise output (recall that this is the same as that for signal-plus-noise). In a standard square-law detector, the standard deviation of the noise-only hypothesis σ_N is given by (see reference 2)

$$\sigma_N = \sqrt{\frac{1}{\pi T} \int_{-\infty}^{+\infty} |H_0(\omega)|^4 V_N^2(\omega) d\omega},$$

(2)

where $H_0(\omega)$ is the pre-detection filter transfer function, $V_N(\omega)$ is the beamformer output noise spectrum, and T is the averaging time. For the sequel, the pre-detection filter transfer function is assumed to be unity; that is, $H_0(\omega) = 1$. Such an assumption is reasonable in the absence of specific knowledge of the source signal. Furthermore, the averaging time is assumed to be large compared to the correlation time of the beamformer output noise. The mean value of the noise-only hypothesis is given by (see reference 2)

$$\mu_N = \frac{1}{2\pi} \int_{-\infty}^{+\infty} |H_0(\omega)|^2 V_N(\omega) d\omega,$$

(3)

with a similar expression holding for the signal-plus-noise mean value. The statistical independence of signal and noise implies that $V_S(\omega) = V_{SN}(\omega) - V_N(\omega)$, so that the deflection coefficient (equation (1)) reduces (in the case of a unity pre-detection filter) to the following:

$$d = \sqrt{\frac{T}{4\pi}} \frac{\int_{-\infty}^{+\infty} V_S(\omega) d\omega}{\left[\int_{-\infty}^{+\infty} V_N^2(\omega) d\omega \right]^{1/2}}.$$

(4)

The goal of the passive sonar detection problem is now restated as a maximization of the deflection coefficient (equation (4)) under the signal and noise conditions of interest.

In the passive sonar application, the signal is unknown, yet the array is steered to a prescribed direction θ. Thus, the source is modeled as a plane wave arriving from direction θ with unknown spectrum $S(\omega)$. Consider a passive array of M sensors with a conventional delay-and-sum beamformer. Let $\mathbf{k}_S$ be the corresponding wavevector of the plane wave from direction θ. The beamformer output power spectrum in the direction θ is represented by the autocorrelation of the time output of the delay-and-sum beamformer. Specifically, this is given by

$$V(\omega) = \int_{-\infty}^{+\infty} E[v(t)\, v(t+\tau)]\, e^{-i\omega\tau}\, d\tau,\tag{5}$$

where the beamformer output voltage $v(t)$ is given by

$$v(t) = \sum_{m=1}^{M} w_m\, u_m(t - \tau_m),\tag{6}$$

with $u_m(t)$ representing the time-domain output of the m-th sensor and τ_m representing the appropriate time delay to match the wavevector $\mathbf{k}_S$.

For the case of sensors receiving noise-only input, the beamformer output power spectrum in equation (5) has been shown (reference 3) to reduce to

$$V_N(\omega) = \mathbf{W}^T \mathbf{U}^*(\omega)\mathbf{M}(\omega)\mathbf{U}(\omega)\mathbf{W},\tag{7}$$

where superscript T represents transpose, $(\cdot)^*$ represents the complex conjugate transpose, and $\mathbf{W}$ and $\mathbf{U}(\omega)$ are given by the following:

$$\mathbf{W} = [w_1, w_2, \cdots, w_M]^T,\tag{8}$$

and

$$\mathbf{U}(\omega) = \mathrm{Diag}\{\exp(i\,\mathbf{k}_S \cdot \mathbf{x}_1),\, \exp(i\,\mathbf{k}_S \cdot \mathbf{x}_2), \cdots, \exp(i\,\mathbf{k}_S \cdot \mathbf{x}_M)\}.\tag{9}$$

The matrix $\mathbf{M}(\omega)$ is a matrix of sensor noise measurement cross-correlations; that is, the (i,j)-th component of $\mathbf{M}(\omega)$ is given by the cross-correlation of the noise output of sensor i with sensor j. For the case of sensors receiving signal-only input with spectrum $S(\omega)$ in the form of a plane wave in direction θ, the beamformer output power spectrum in equation (5) has been shown (reference 4) to reduce to

4

$$V_S(\omega) = S(\omega) \, G \left(\sum_{m=1}^{M} d_m(\theta) \, w_m \right)^2, \tag{10}$$

where G is the power gain of an individual sensor and $d_m(\theta)$ is the directivity loss of the m-th sensor when receiving a signal in the direction θ. For example, omnidirectional sensors have $d_m(\theta) \equiv 1$, and cosine-directive velocity sensors have

$$d_m(\theta) = \begin{cases} (\mathbf{k}_S(\theta) \cdot \mathbf{n}_m)/k_0, & \text{for } \mathbf{k}_S(\theta) \cdot \mathbf{n}_m > 0 \\ 0, & \text{otherwise} \end{cases}, \tag{11}$$

where $\mathbf{n}_m$ is the sensor outward normal and k_0 is the free-space wavenumber. Equations (7) and (10) provide the power spectra required for general evaluation of the deflection coefficient expressed in equation (4). Together, these equations represent the analytical form of the conventional passive sonar array performance.

The goal of conventional beamformer shading weight optimization is to maximize the performance (as measured by the deflection) through the adjustment of the shading weights w_m. This is accomplished by minimizing the denominator of the deflection while holding the value of the numerator fixed. This problem is stated in the form of a nonlinear optimization problem as follows:

$$\min_{\mathbf{W}} \int_{-\infty}^{+\infty} V_N^2(\omega) \, d\omega$$

such that $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ (12)

$$\sum_{m=1}^{M} d_m(\theta) \, w_m = 1.$$

The constraint expression in (12) is a direct result of holding the beamformed signal power output expression (10) constant with respect to the available parameters w_m. Note that the sensor gain G and the signal spectrum $S(\omega)$ are not available as degrees of freedom in the optimization and, therefore, are taken as unknown constant parameters with respect to the optimization problem (12). The optimization problem (12) is similar in intent to the goal of Minimum Variance Distortionless Response (MVDR) beamforming, but the optimization problem varies in two important respects. First, the weights are constrained to be real and, second, a single set of weights is found over the entire frequency band of interest. These differences make the nonlinear programming approach amenable to the conventional beamformer design and thus it is not an "adaptive beamformer" approach, although the resulting weights are adaptively determined. In practice, the limits of integration in (12) are restricted to the expected frequency band of interest, thus providing a limiting case (as bandwidth goes to zero) of the frequency domain beamformer optimization problem that is solved in MVDR.

3. SOLVING THE OPTIMIZATION PROBLEM USING EXISTING NUMERICAL TECHNIQUES

The numerical optimization of nonlinear objective functions under constraints (the nonlinear programming problem) is a well-studied problem with many available solution techniques. Unfortunately, such problems are often prone to multiple local minima, and numerical methods that do not start near the solution may converge to one of these locally optimal solutions with no warning. For the special case of nonlinear programming problems that are convex optimization problems, all local minima are guaranteed to be global minima, so the problem of local solutions no longer exists. For these problems, the quadratic programming (QP) and sequential quadratic programming (SQP) methods are often utilized due to their rapid convergence properties and ease of implementation. QP methods are applicable only to a further subset of problems in which the objective is quadratic, whereas SQP methods apply a process of sequentially approximating the problem as a quadratic objective in a convergent manner. In this section, the utility of conventional SQP methods in solving the optimization problem (equation (12)) is examined. A thorough overview of SQP methods is given in reference 5.

The objective function and simple linear constraints shown in equation (12) are a convex optimization problem. By definition, an optimization problem is convex if the objective function is a convex function and the set of feasible points (the set of variables that meet the constraints) is convex. For the problem under consideration (i.e., equation (12)), the objective is convex since it is the square of a quadratic form (equation (7) shows that $V_N(\omega)$ is a quadratic form in the weights $\mathbf{W}$), and the constraint set is a convex linear combination of the variables. Thus, problem (12) is a convex nonlinear programming problem and is thus suitable for solution using SQP methods.

Consider a general nonlinear programming problem as an objective function and constraints as follows:

$$\min_{x} f(x)$$

such that (13)

$$C(x) \leq 0.$$

Any general nonlinear programming problem can be placed in this form by incorporating all equality constraints into the objective function. The Lagrangian function of the optimization problem (13) combines the objective and constraints into a single function $L(x,\lambda)$, where λ represents the Lagrange multipliers associated with (13). The general SQP method considers a sequence of estimates of x (given by x_k), where at each estimate a Taylor series expansion of the objective is used to obtain the quadratic function

$$f(x_{k+1}) \approx f(x_k) + \nabla f(x_k)^T d_k + \tfrac{1}{2} d_k^T \nabla^2 L(x_k, \lambda_k) d_k, \qquad (14)$$

7

where

$$d_k = x_{k+1} - x_k \tag{15}$$

represents the step taken to get to the next iterate. It can be shown (see reference 6) that this iteration scheme, when convergent, leads to a solution that satisfies the first and second order conditions for optimality (the Karush-Kuhn-Tucker conditions). The convergence is guaranteed whenever the Hessian of the Lagrangian maintains positive-definiteness. The numerical minimization of quadratic equation (14) is easily performed using standard QP solution techniques (see reference 7).

Most commercial general-purpose optimization codes for nonlinear programming solve equation (14) iteratively. The computational savings is that a single algorithm easily handles general objective functions and constraints. This makes it appropriate for general-purpose codes. However, the user must still define the objective function, the gradient function, and the Hessian in a manner that is suitable for the general-purpose solver. One of the most reliable of the general-purpose SQP solvers is NPSOL (reference 8) from Stanford University's Systems Optimization Laboratory. NPSOL solves equation (14) for general objective functions and general constraints. The user must provide a separate subroutine to evaluate the objective function and the gradient at an arbitrary iteration step. The Hessian of the Lagrangian is evaluated approximately using a Broyden Fletcher Goldfarb Shanno (BFGS) update formula (see reference 9). Such an update provides a computationally reliable technique to improve estimates of the Hessian.

A numerical implementation of array shading weight optimization using NPSOL was employed as the first component of this effort. The results of this algorithm are found in references 4 and 10. For the problem under consideration (equation (12)), the required objective function and gradient function are written as

$$f(\mathbf{W}) = \int_{\omega_0}^{\omega_1} \left(\mathbf{W}^T \mathbf{U}^*(\omega) \mathbf{M}(\omega) \mathbf{U}(\omega) \mathbf{W} \right)^2 d\omega, \tag{16}$$

and

$$\nabla f(\mathbf{W}) = 4 \int_{\omega_0}^{\omega_1} \left(\mathbf{W}^T \mathbf{U}^*(\omega) \mathbf{M}(\omega) \mathbf{U}(\omega) \mathbf{W} \right) \mathbf{U}^*(\omega) \mathbf{M}(\omega) \mathbf{U}(\omega) \mathbf{W} \, d\omega, \tag{17}$$

where the integrals are evaluated using the method of overlapping parabolas, and the frequency limits of $[\omega_0, \omega_1]$ represent the limiting band of the sonar system design. The numerical implementation of this objective within NPSOL worked well for small arrays. As the number of elements (size of $\mathbf{W}$) grew, the performance of NPSOL degraded. In particular, the ability of NPSOL's BFGS algorithm to handle the updating of such a large Hessian (the Hessian size is $M \times M$ for an M-element array problem) faulted. In particular, for arrays larger than 100 elements, the Hessian update algorithm actually diverged, producing a total lack of convergence. Numerical investigations showed that the problem is inherent in using a BFGS update of the Hessian, and a method without such a requirement was sought.

8

An alternative approach to the use of a general solver for the solution of the QP subproblem (14) is to develop a specialized algorithm that lacks the flexibility to handle general objectives but is tuned so that the Hessian is included exactly. The main difficulty with including the Hessian directly is that the Lagrangian is usually much more complex than the simple objective. By limiting attention to the array shading problem (12), the constraints are all linear; hence, the Hessian of the objective is identical to the Hessian of the Lagrangian (all second derivatives of constraint terms are necessarily zero). Thus, for the specific array shading optimization problem, the Hessian is explicitly available as

$$
\begin{aligned}
\nabla^2 f(\mathbf{W}) \;=\; & 4 \int_{\omega_0}^{\omega_1} \mathbf{U}^*(\omega)\mathbf{M}(\omega)\mathbf{U}(\omega)\big(\mathbf{W}^T\mathbf{U}^*(\omega)\mathbf{M}(\omega)\mathbf{U}(\omega)\mathbf{W}\big)d\omega \\
+ \; & 8 \int_{\omega_0}^{\omega_1} \mathbf{U}^*(\omega)\mathbf{M}(\omega)\mathbf{U}(\omega)\mathbf{W}\mathbf{W}^T\mathbf{U}(\omega)\mathbf{M}(\omega)\mathbf{U}^*(\omega)d\omega,
\end{aligned}
\tag{18}
$$

which is readily computed at each iteration. A general-purpose nonlinear programming solver does not allow such details to be input by the user. By developing a specialized code that solves only the array shading weight optimization problem, the numerical issues associated with approximation updates to the Hessian will no longer be a concern, since the Hessian will be evaluated directly at each step. It is noted that such an approach is not advisable for small problems, since the evaluation of (18) has considerable computational costs relative to applying the computationally efficient BFGS method.

4. ARRAY SHADING-SEQUENTIAL QUADRATIC PROGRAMMING (AS-SQP) ALGORITHM

4.1 MATHEMATICAL DESCRIPTION

The Array Shading-Sequential Quadratic Programming (AS-SQP) algorithm was developed to provide robust numerical solutions to the specific array shading optimization problem (12) in cases where general-purpose solvers fail to converge. The mathematical derivation of AS-SQP is developed for a general convex objective function with a single convex linear equality constraint and simple bound constraints on the variables (array weights). Application to the problem (12) is accomplished by evaluating expressions (16), (17), and (18) to the appropriate components of the algorithm. Consider the following nonlinear optimization problem:

$$
\min_{\mathbf{w} \in R^M} F(\mathbf{w})
\tag{19a}
$$

subject to the linear equality constraint

$$
\sum_{m=1}^{M} w_m = 1
\tag{19b}
$$

and the lower-bound and upper-bound constraints on the array weights

$$0 \le w_m \le 1, \quad m = 1, 2, ..., M.$$

(19c)

In the above formulas, the array weights are expressed in terms of the vector $\mathbf{w} = [w_1, w_2, ..., w_M]^T$. This formulation is consistent with the nonlinear programming problem formulated in the preceding section. Note that non-unity element contributions to the steer direction (i.e., directivity effects) can be accounted for by adjusting the definition of "weight" to include the conventional weight combined with the directive component in the steered direction (i.e., $w_m \leftarrow d_m(\theta)w_m$). For simplicity of exposition, the case of unity directivity (omnidirectional sensors) is considered in this mathematical derivation; however, the computational implementation includes the directivity effects.

Generally, an SQP algorithm involves two parts: the determination of the search direction δ obtained from the solution of a QP subproblem and the determination of the step size along the search direction δ. At the end of each iteration, the weight vector $\mathbf{w}$ is updated and the procedure is repeated until either a convergent result is obtained or the maximum number of allowed iterations is reached. At each iteration of the SQP algorithm, the gradient $\mathbf{g}$ and the Hessian $\mathbf{H}$ of the objective function $F(\mathbf{w})$ are evaluated at the current iterate $\mathbf{w}^{(k)}$, where k is the iteration index. The search direction $\delta^{(k)}$ at the point $\mathbf{w}^{(k)}$ is then determined through the solution of a QP subproblem. The subproblem involves the minimization of the quadratic function approximation $Q^{(k)}$ to the objective function at the point $\mathbf{w}^{(k)}$ subject to constraints (19b) and (19c).

The AS-SQP algorithm is explicitly focused on the problem of optimal shading weights. Thus, rather than a generic set of constraints, only linear shading weights of the form (19b) are considered (along with bound constraints on the individual weights). The QP subproblem for AS-SQP is given as

$$\min_{\delta \in R^M} Q^{(k)}(\delta)$$

(20a)

subject to the linear equality constraint

$$\sum_{m=1}^{M} \delta_m = 0$$

(20b)

and the lower- and upper-bound constraints on the components of the search direction vector

$$-1 \le \delta_m \le 1, \quad m = 1, 2, ..., M.$$

(20c)

In expression (20a), the objective function is defined as

$$Q^{(k)}(\delta) = \tfrac{1}{2}\delta^T \mathbf{H}^{(k)}\delta + \delta^T \mathbf{g}^{(k)} + F(\mathbf{w}^{(k)}),$$

(21)

10

where $\mathbf{H}^{(k)}$ and $\mathbf{g}^{(k)}$ denote the Hessian and gradient of $F(\mathbf{w})$, respectively, evaluated at the current iterate $\mathbf{w}^{(k)}$. Constraints (20b) and (20c) for the search direction δ are consistent with array weight constraints (19b) and (19c).

To derive constraints (20b) and (20c) in the QP subproblem, the search direction vector is expressed as

$$\delta = \mathbf{w} - \mathbf{w}^{(k)}. \tag{22}$$

The components of δ are given as

$$\delta_m = w_m - w_m^{(k)}, \quad m = 1, 2, \ldots, M. \tag{23}$$

If the M components of equation (23) are summed, linear constraint (20b) is obtained; i.e.,

$$\sum_{m=1}^{M} \delta_m = \sum_{m=1}^{M} w_m - \sum_{m=1}^{M} w_m^{(k)} = 0, \tag{24}$$

where expression (19b) has been applied to the sums involving $\mathbf{w}$ and $\mathbf{w}^{(k)}$. Lower- and upper-bound constraints (20c) of the components of the search direction vector can be derived from formulas (19c) and (23). From constraint (19c), one has

$$-1 \le w_m - w_m^{(k)} \le 1, \quad m = 1, 2, \ldots, M. \tag{25}$$

Substitution of expression (23) into equation (25) yields the variable constraints (20c) in the QP subproblem.

Once the solution to the QP subproblem (20) is obtained, the new iterate $\mathbf{w}^{(k+1)}$ is given by

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \alpha^{(k)} \delta^{(k)} \tag{26}$$

where the non-negative scalar $\alpha^{(k)}$ is the step size to be determined. In the QP subproblem (20), because the quadratic function approximation to $F(\mathbf{w})$ at $\mathbf{w} = \mathbf{w}^{(k)}$ is minimized at $\mathbf{w} = \mathbf{w}^{(k)} + \delta^{(k)}$, this minimum corresponds to a step size of $\alpha^{(k)} = 1$ in expression (26). However, because the components of $\mathbf{w}^{(k+1)}$ must lie within the constraint region (19c), limitations are placed on $\alpha^{(k)}$.

The second part of the SQP algorithm involves the determination of the step size $\alpha^{(k)}$ in expression (26). The components of the new iterate $\mathbf{w}^{(k+1)}$ must satisfy constraints (19c); i.e.,

$$0 \le w_m^{(k)} + \alpha^{(k)} \delta_m^{(k)} \le 1, \quad m = 1, 2, \ldots, M. \tag{27}$$

In addition, the step size must satisfy

$$0 \le \alpha^{(k)} \le 1. \tag{28}$$

Inequality (27) places constraints on the step size $\alpha^{(k)}$ in order to ensure that the components of $\mathbf{w}^{(k+1)}$ stay within the constraint space. Because the new iterate $\mathbf{w}^{(k+1)}$ must satisfy the equality constraint (19b), the upper bound of inequality (27) will never be reached. The mth component of the argument in expression (27) will reach or exceed its lower-bound constraint for the following step sizes:

$$\alpha^{(k)} \ge -\frac{w_m^{(k)}}{\delta_m^{(k)}}, \quad \delta_m^{(k)} > 0, \quad m = 1, 2, \ldots, M, \tag{29}$$

$$\alpha^{(k)} \le -\frac{w_m^{(k)}}{\delta_m^{(k)}}, \quad \delta_m^{(k)} < 0, \quad m = 1, 2, \ldots, M. \tag{30}$$

Condition (29) is trivial because $\alpha^{(k)}$ is already restricted to the interval [0,1]. Therefore, the step size $\alpha^{(k)}$ in the optimization problem (27) is determined from the components of $\mathbf{w}^{(k+1)}$ with negative search direction components. Define the set I_k as follows:

$$I_k = \left\{ m : \delta_m^{(k)} < 0 \right\}. \tag{31}$$

Therefore, from formulas (30) and (31), the maximum allowable step size at the kth iteration is given by

$$\alpha^{(k)} = \min_{m \in I_k} \left(-\frac{w_m^{(k)}}{\delta_m^{(k)}} \right). \tag{32}$$

The AS-SQP algorithm also contains a user-defined step size α_u, which serves as the default step size during each major iteration, where $0 < \alpha_u \ll 1$. The user-defined step size is small because the gradient and Hessian of $F(\mathbf{w})$ will vary as $\mathbf{w}$ moves along the search direction. At each major iteration, the step size is determined as

$$\alpha^{(k)} = \min \left\{ \alpha_u, \min_{m \in I_k} \left(-\frac{w_m^{(k)}}{\delta_m^{(k)}} \right) \right\}. \tag{33}$$

At the end of each iteration, the components of the updated weight vector $\mathbf{w}^{(k+1)}$ are checked to see if they are active (i.e., to see if any lie on the lower boundary of the constraint region (19c)). For each active component of $\mathbf{w}^{(k+1)}$, the lower bound of the corresponding search direction component is set to zero (i.e., $0 \le \delta_m^{(k+1)} \le 1 \quad \forall\, m \ni w_m^{(k+1)} = 0$). The QP subproblem is then recalculated with the updated weight vector and direction constraints. The above procedure is repeated until either a convergent result is obtained or the maximum number of iterations is

reached. A complete flowchart of the AS-SQP numerical algorithm is presented in the appendix. Note that the addition of non-unity directivity complicates the expressions somewhat; however, the formulation is identical to that described above.

4.2 COMPUTATIONAL IMPLEMENTATION

The numerical implementation of the AS-SQP algorithm has been performed as a software program called NumShade (abbrevation for Numerical Array Shading Weight Optimization). NumShade was written in FORTRAN 95 with a Windows-based graphical user interface (GUI), using Lahey FORTRAN 95 version 5.7 along with Lahey's Winteracter development environment. The entire package (software with GUI) is available as a single Windows executable file that will run on any computer under the Windows operating system, provided there is enough memory available on the computer for the calculations. Dynamic memory allocation within FORTRAN 95 was employed to improve this portability. Thus, computer systems with little memory will suffice for solving small array problems (few elements and/or low bandwidth) and a larger memory computer can be employed to solve a larger array shading problem. This scalability of problem size is inherent in the software structure and, thus, is transparent to the user.

As part of the computational implementation of the AS-SQP algorithm, expressions for the objective function and its gradient and Hessian were derived and evaluated directly in the software. This eliminates the need for numerical methods to approximate the derivatives and thus removes one of the major drawbacks of using standardized numerical software. These evaluations are accomplished in a secondary subroutine that evaluates expressions (16), (17), and (18) at a given iteration using the method of overlapping parabolas to provide the numerical evaluation of the integrals. The GUI of the software is shown in figure 1. A second software program that employs NPSOL (for use with small array problem) is also available with an identical GUI. The *INPUTS* side of the GUI takes in filenames for element position data, element normal data, and noise cross-correlation data. These filenames may be typed in, or the appropriate search button can be used to open a standard Windows file-chooser box. All of the input files are ASCII text files that contain the data specific to an array. The *Sensor Locations* file consists of $M + 1$ lines, the first line containing the number of elements, and the remaining lines containing the (x, y, z) positions of the elements (separated by spaces). The *Sensor Normals* file is identically formatted, except that the positions are replaced with the (x, y, z) components of the unit outward normal to each sensor. This information is used to determine the sensor directivity response. The default version of the software assumes cosine-directive elements; other element directivity patterns are readily incorporated in special releases of the software. The *Noise Cross-Corr* file contains the element-to-element noise cross-correlation for each frequency of interest. This file is by far the largest file needed to run the program. The formatting of the file is as follows: the first line is frequency 1 value and the next line is $2*M$ numbers representing the frequency 1 component of the complex noise cross-correlation of sensor 1 with each other sensor, in order. The complex components are represented in pairs of real component followed by imaginary component, with a space between the two, then another space before the next complex pair. The successive lines continue this cross-correlation for each of the sensors until frequency 1 is exhausted; then, the process is repeated for each frequency.

The resulting file contains $F*(M+1)$ rows, where F is the number of frequencies and M is the number of elements.
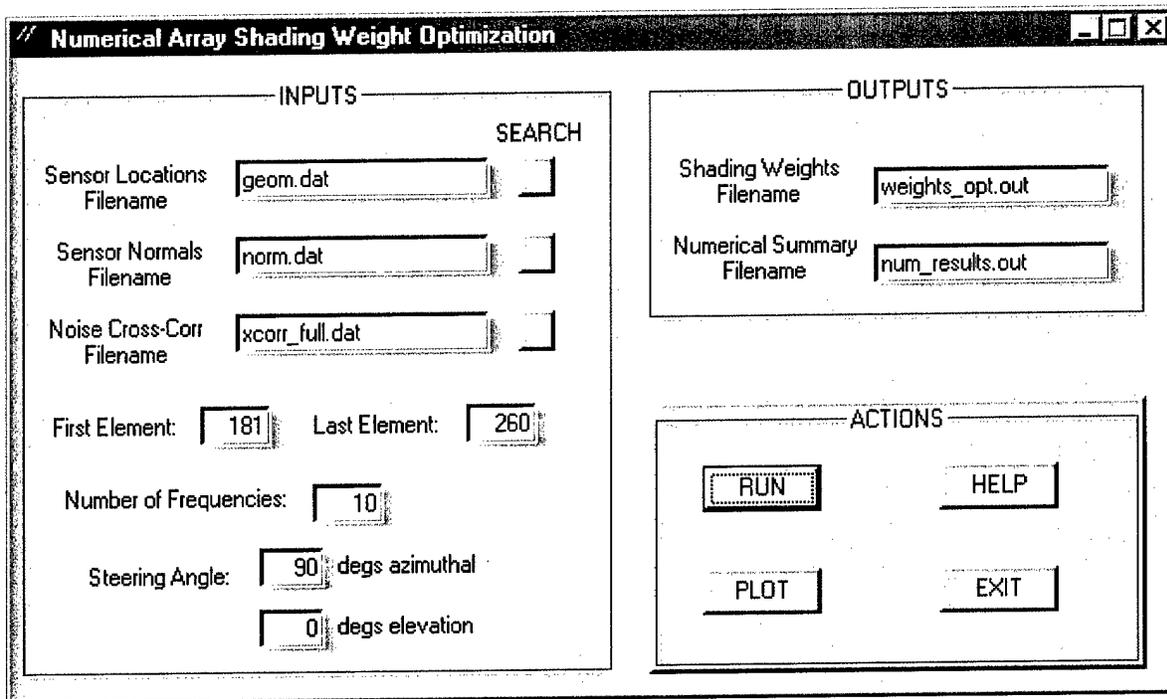
Figure 1. Graphical User Interface for the Optimization Software

The user has the option of selecting a subset of the sensor elements (or subarray) by specifying a first and last element. This allows the entire database of noise information to be stored in a single ASCII text file, and the software accesses the components necessary for the optimization that is desired. Furthermore, the number of frequencies can be specified as less than the number available. At this time, the frequency list always begins with the first frequency in the noise cross-correlation file. The final setup input is the array steering angle, which assumes a traditional submarine orientation with (azimuthal, elevation) pairs of (0°,0°) pointing along the negative x-axis, (90°,0°) pointing along the y-axis, and (0°,90°) pointing along the z-axis. A final user input is the selection of output filenames. The *Shading Weights* file contains two columns—element number and final shading weight. The *Numerical Summary* file contains summaries of the numerical performance of the algorithm, which is used by advanced users for assessing convergence performance.

Once the files are set up and the appropriate fields are all filled in on the GUI, the user presses the *RUN* button. After the data files are loaded into memory, a secondary warning window pops up to inform the user that the run is set up. When the *OK* button on that window is pressed, the optimization calculation begins. The output data files can be read in to any offline program of the user's choosing. There is a very basic plotting facility under the *PLOT* button that will display a window with the nominal (unity shading) objective and the resulting

14

optimal objective plotted over the frequency band of the calculation. It is not recommended that this plotting be used for anything other than "quick-looks" at the results to determine that the program is functioning properly. The *HELP* button contains contact information for the author. The *EXIT* button closes the program window.

5. NUMERICAL RESULTS

A set of test data from the WHITEFISH structural acoustic model was taken under a conformal velocity sonar array test in 1996 at the Intermediate Scale Measurement System (ISMS). This data set was collected to address many issues, including a high spatial resolution measurement of the structural noise field on a large-scale array model. For the purposes of the array shading optimization algorithm, the data set provides a convenient mechanism for testing the array shading algorithms on a large array with a reasonable level of complexity in the noise field. The test is described in detail in reference 10 and the reader is referred there for specific details of the experimental setup. The test setup of the WHITEFISH model for array data collection is shown in figure 2. The model contains a series of both small and deep frames within a steel pressure hull. A sequence of 25 shakers were employed along the model to provide very detailed structural acoustic forcing patterns over a nondimensional frequency range of $0.5 \leq k_0 a \leq 30$, where k_0 is the wavenumber in the fluid and a is the nominal radius.
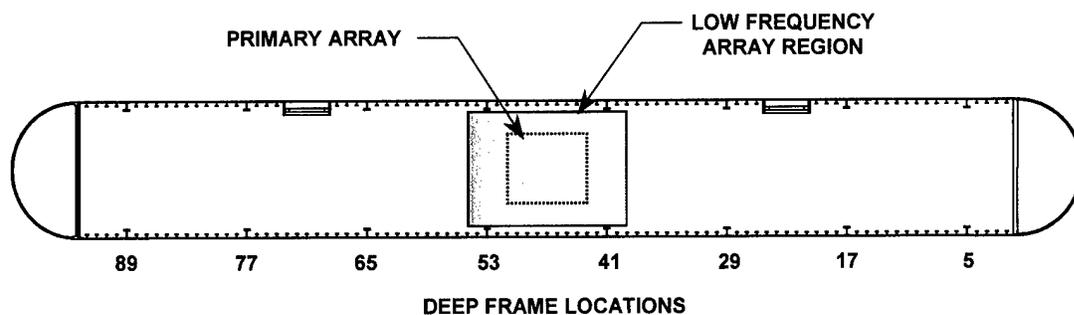
Figure 2. Array Geometry for the Test Data Set

The WHITEFISH model hull was coated with a pressure-release material, and conformal velocity sensors were placed in two array configurations as shown in figure 2. The data used to test the optimization algorithms come from the *primary array* region, which consists of a patch of 480 regularly-spaced elements as shown in figure 3. The elements that are colored red were considered faulty elements and were removed from the analysis. The primary area of interest in this numerical study is subarrays of varying sizes beginning at element 181 and continuing until element 380. This provided arrays of sizes 1 x 20 up to 10 x 20 for analysis purposes. Further-

15

more, to reduce the size of data files, only a subset of the frequency band with bandwidth of approximately $k_0 a = 3.2$ was used in the numerical computations. Examination of the data showed that this limited frequency band was sufficient to capture the complexity of the noise field and thus illustrate the array shading weight optimization algorithm.

■	■	■	61	81	101	121	141	161	181	201	221	241	261	281	301	321	■	361	381	401	■	■	■
■	22	42	62	82	102	122	■	162	182	202	222	242	262	282	302	322	342	362	382	402	422	442	■
	23	43	63	83	103	123	143	163	183	203	223	243	263	283	303	323	343	363	383	403	423	443	■
4	24	44	64	84	104	124	144	164	184	204	224	244	264	284	304	324	344	364	384	404	424	444	464
5	25	45	65	85	105	125	145	165	185	205	225	245	265	285	305	325	345	365	385	405	■	445	465
6	26	46	66	86	106	126	146	166	186	206	226	246	266	286	306	326	346	366	386	406	426	446	466
7	27	■	67	87	107	127	147	167	187	207	227	247	267	287	307	327	347	367	387	407	427	447	467
8	28	48	68	88	108	128	148	■	188	208	228	248	268	288	308	328	348	368	388	408	428	448	468
9	29	49	69	89	109	129	149	169	189	209	229	249	269	289	309	329	349	369	389	409	429	449	469
10	30	50	■	90	110	130	150	170	190	210	230	250	270	290	310	330	350	370	390	410	430	450	470
11	31	51	71	91	111	131	151	171	191	211	231	251	271	291	311	331	351	371	391	411	431	451	471
12	32	52	72	92	112	132	152	172	192	212	232	252	272	292	312	332	352	372	392	412	432	452	472
13	33	53	73	93	113	133	153	173	193	213	233	253	273	293	313	333	353	373	393	413	433	453	473
14	34	54	74	94	114	134	154	174	194	214	234	254	274	294	314	334	354	374	394	414	434	454	474
15	35	55	75	95	115	135	155	175	195	215	235	255	275	295	315	335	355	375	395	415	435	455	475
16	36	56	76	96	116	136	156	176	196	216	236	256	276	296	316	336	356	376	396	416	■	456	476
17	37	57	77	97	117	137	157	177	197	217	237	257	277	297	317	337	357	377	397	■	437	457	477
■	38	58	78	98	118	138	158	178	198	218	238	258	278	298	318	338	358	378	398	418	438	458	■
■	39	59	79	99	119	139	159	179	199	219	239	259	279	299	319	339	359	379	399	419	439	459	■
■	■	■	80	100	120	140	160	180	200	220	240	260	280	300	320	340	360	380	400	420	■	■	■

Figure 3. Map of the Sensor Locations Used in the Test Array

16

Because NPSOL was shown to perform well on small arrays (see section 3), a subarray of 80 elements (from numbers 181-260) was used to compare the convergence of the AS-SQP algorithm with NPSOL. This 4 x 20 array provides enough aperture that noticeable benefit is expected from optimizing shading weights based on noise measurements. The performance of each algorithm is considered by measuring the relative improvement in broadband SNR relative to that achieved by a unity-shaded array over the same frequency band. The use of SNR as a performance metric is natural since that is a design goal of any array. For the passive array, the signal is unknown; however, since the optimization algorithm constrains the signal gain to be constant (see equation (12)), the resulting ratio of SNRs (or SNR improvement) is independent of the signal level. Thus, the improvement in SNR relative to unity shading is the primary performance metric for the shading algorithms. The result of the convergence versus iteration number for the two algorithms is shown in figure 4. For this computation, a frequency bandwidth of $k_0 a = 2.5$ was used.
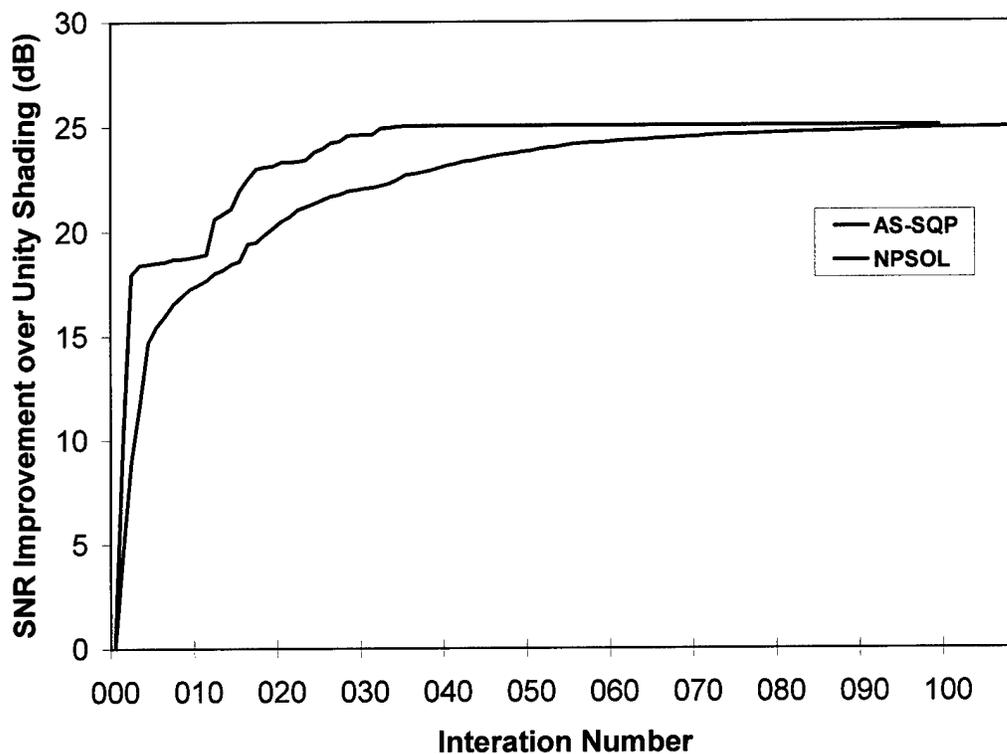
Figure 4. Algorithm Convergence Measured by SNR Improvement Over Unity Shading

From figure 4, it is clear that both algorithms converge to a similar solution (and converge to the same amount of improvement over unity shading). Both algorithms converge uniformly (they always improve) as is expected for any SQP algorithm applied to a convex nonlinear programming problem. Also, the AS-SQP converges more rapidly initially, yet the NPSOL algorithm converges at a steadier rate. This is attributed to the Hessian evaluation strategies employed. Since NPSOL continually re-estimates the Hessian in an iterative fashion, it is

17

limited to incremental performance improvement at each iteration. AS-SQP, on the other hand, re-computes the Hessian from "scratch" at each iteration, thus greatly improving convergence when there is a direction of rapid change to be followed to the next iteration. NPSOL smoothes out these rapid changes by only allowing gradual shifts to the Hessian, thus providing slower, more stable convergence. While the number of iterations implies that AS-SQP converges more rapidly, each iteration is much more expensive, so that AS-SQP actually converges (in wall-time) much more slowly than NPSOL. This is to be expected, since the evaluation of the Hessian via equation (18) is the most computationally intensive part of either algorithm. Since both algorithms provide similar results and run-time is not considered in this study, AS-SQP was used in the remaining calculations due to its ability to handle arbitrarily large array shading problems.

The tremendous performance improvement shown in figure 4 leads to the question of whether the bandwidth was large enough to consider this a true broadband performance result. For a broad enough frequency band, any complexity in the noise will be seen by the array as white noise (due to the frequency integration) and, thus, the unity shading should provide near-ideal performance. To answer the bandwidth question, an examination of the performance improvement relative to frequency bandwidth was conducted using the same 4 x 20 subarray. The results of both decreasing and increasing the bandwidth are plotted in figure 5. From the plot, it is clear that, although the performance improvement is degrading as bandwidth increases, the rate of decrease is slow enough that substantial gains can be expected for very large bandwidth problems. Also, note that the limit as bandwidth goes to zero approaches a value of 28 dB, which is the limit of performance of the delay-and-sum beamformer for that size of array (superdirectivity or non-real weights are the only ways to improve performance beyond that limit). Thus, the array shading weight optimization software can be employed to set performance limits.
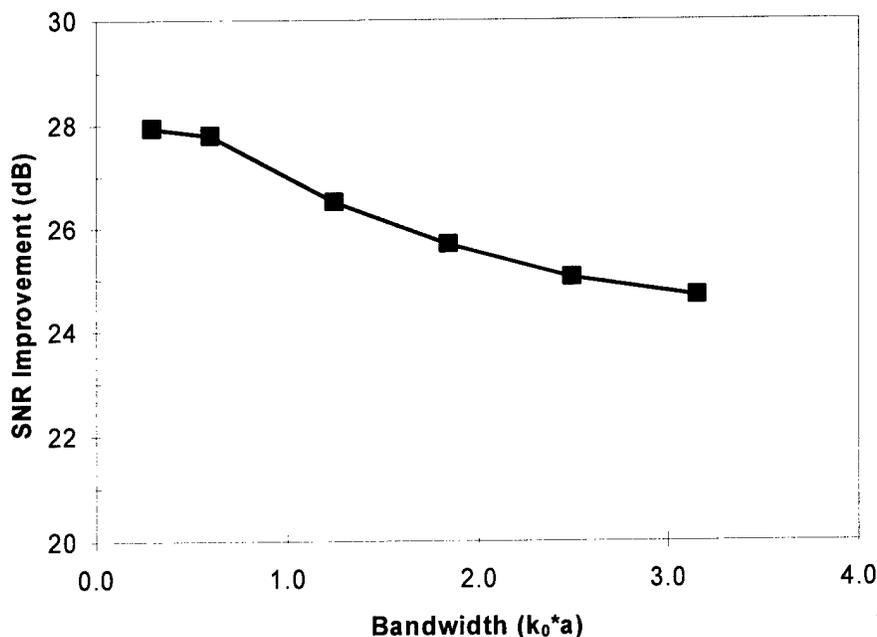

Figure 5. Performance Improvement as a Function of Frequency Bandwidth

In addition to the effect of performance degradation due to increased frequency, there is also an expected change in performance due to changes in spatial extent (aperture size). Since an array functions as a spatial filter, eventually the array size becomes large enough that the noise response appears spatially white. This phenomenon is clearly illustrated in figure 6, where a variety of aperture sizes (all beginning with element number 181) were considered. All of the calculations for this plot used the $k_0a = 2.5$ bandwidth. Note that the AS-SQP algorithm is effective even on arrays of 200 elements (twice the size of the array optimization problem that the NPSOL algorithm handles). For large arrays (more than 100 elements), the performance improvement degrades with array size, as the spatial filtering effectiveness of the aperture becomes a dominant mechanism. This illustrates another area of array analysis where the optimal shading algorithm is useful—the setting of array sizes to take advantage of the spatial filtering effect. For the smaller apertures (20, 40, and 60 elements), the array width is so small (all of these have 20-element columns, so the widths are 1, 2, and 3 elements, respectively) that the array shading cannot be used effectively. Thus, there is a minimum effective aperture size before array shading optimization is merited. This size is dependent on both the frequency bandwidth and the specific structure of the noise profile.
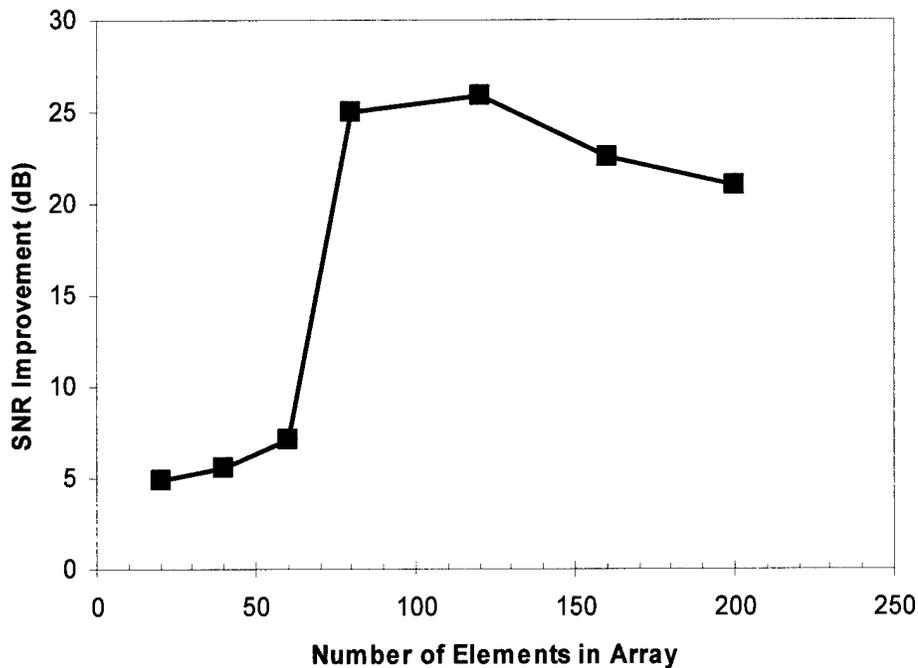
Figure 6. Performance Improvement as a Function of Array Size

Since the array shading optimization algorithms are designed to minimize the noise response $V_N(\omega)$ while simultaneously maintaining signal performance $V_S(\omega)$, it is expected that the specific structure of the noise field has a large impact on the resulting optimal shading weights. To examine the impact of the specific structure of the noise field on the algorithm's performance, a comparison of optimal shading weights obtained from a simulated array with white noise

(figure 7) was computed and compared to that from a section of the test array with an identical geometry (figure 8). In both of these cases the resulting shading weights were scaled for visualization so that the maximum weight is at unity. The white noise case in figure 7 shows that the array shading algorithm functions as expected, providing a standard tapered pattern in the shading distribution. The measured noise case takes advantage of the element-level noise readings and places higher weights on less-noisy elements while attempting to still distribute the weights across the entire aperture to maintain array gain. This notion of array gain is not an added feature—the algorithm is only solving the standard array shading optimization problem of equation (12).

To see how much the optimal shading weight selection depends on local noise versus array characteristics, the array from figure 8 was extended to double its size and the new shading weights were computed. The results are shown in figure 9, where the left half of the array consists of the same components as those shown in figure 8 and the same frequency range and noise data were applied in both cases. While there are some elements that are practically "zeroed-out" in both cases, the overall pattern of the optimal shading weight selection changes dramatically. This suggests that selecting the correct array shading weights is more complicated than just removing the noisiest elements or scaling weights inversely with noise level. Rather, the complicated patterns of noise cross-correlations impact the performance of the array as a spatial filter in a complex manner. The ability of the optimization approach to expose those patterns is responsible for the tremendous performance improvements that are garnered.
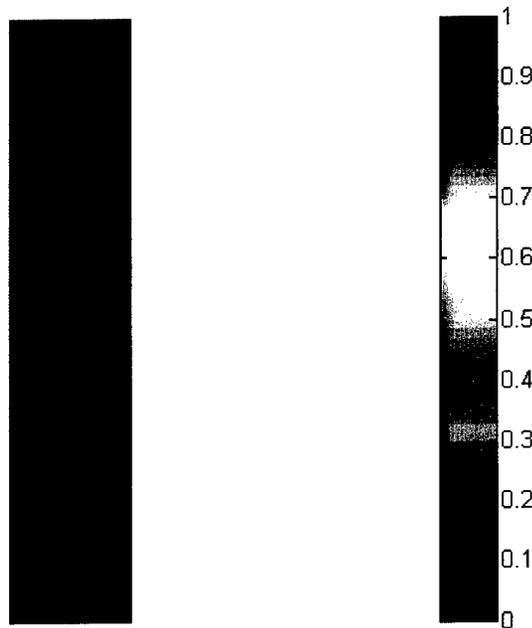

Figure 7. Optimal Shading Weights for 80-Element Array with White Noise
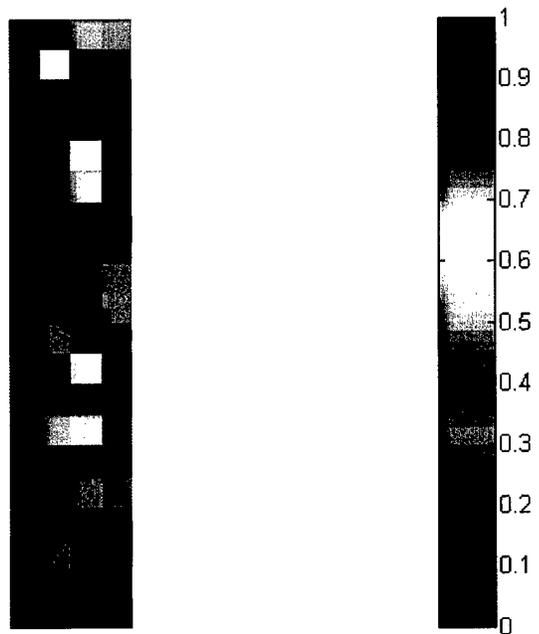
Figure 8. Optimal Shading Weights for Test Array Using Elements 181-260
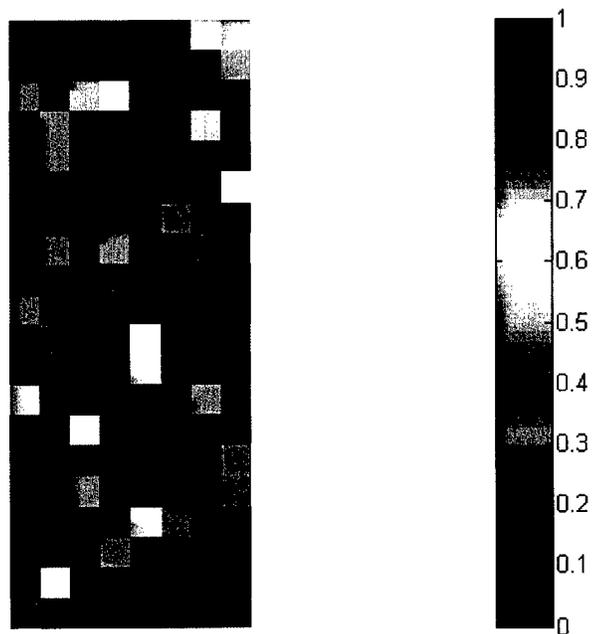
Figure 9. Optimal Shading Weights for Test Array Using Elements 181-340

6. CONCLUSIONS

A numerical optimization algorithm for determining the optimal conventional beamformer array shading weights has been developed. This algorithm is based on maximizing the beamformer deflection coefficient using knowledge of the received noise characteristics of the individual array elements. Two algorithms were developed. The first used a robust off-the-shelf numerical optimization method and proved reliable for small arrays (≤100 elements), while the second algorithm was a customized numerical approach that is specific to the array shading weight optimization problem. The algorithms were tested against test array data and showed tremendous improvement (usually greater than 20 dB) in signal-to-noise ratio relative to a unity shaded array.

The new array shading weight optimization algorithm may be used in a variety of ways. For array designers, the limits of conventional delay-and-sum beamforming can now be adequately explored and compared to frequency-domain adaptive beamforming techniques given the same assumptions (i.e., both approaches take advantage of the same observed noise data). Furthermore, the use of noise models in design arrays with the optimal shading can point out the proper sizing of an array to take advantage of the spatial complexity of the noise field, thus taking full advantage of the natural spatial filtering ability of arrays. Also, the algorithms could provide guidance on the "most important" sections of a given array design by showing which elements receive the largest optimal weights. Finally, a real-time version of the algorithms could be coded to provide an *in situ* re-shading scheme for tactical arrays based on the currently observed noise characteristics. This would greatly improve array performance without impacting the overall structure of the delay-and-sum beamformer.

REFERENCES

1. R. L. Streit and T. A. Wettergren, "An Objective Function for Optimal Hull Array Design Using a Broadband Detection Criterion," NUWC-NPT Technical Memorandum 990124, Naval Undersea Warfare Center Division, Newport, RI, 15 December 1999.

2. H. L. Van Trees, *Detection, Estimation, and Modulation Theory*, Wiley, New York, 1968, Part I, Chapter 2.

3. T. A. Wettergren, J. P. Casey, and R. L. Streit, "A Numerical Optimization Approach to Acoustic Hull Array Design," *Journal of the Acoustical Society of America*, vol. 112, no. 6, 2002, pp. 2735-2741.

4. T. A. Wettergren and C. M. Traweek, "Optimal Determination of Beamformer Shading Weights for Conformal Velocity Sonar," preprint submitted to *IEEE Journal of Oceanic Engineering*.

5. P. T. Boggs and J. W. Tolle, "Sequential Quadratic Programming," *Acta Numerica*, vol. 4, 1996, pp. 1-51.

6. J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999, Chapter 18.

7. J. R. Bunch and L. Kaufman, "A Computational Method for the Indefinite Quadratic Programming Problem," *Linear Algebra and Its Applications*, vol. 34, 1980, pp. 341-370.

8. P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, "User's Guide for NPSOL 5.0: A FORTRAN Package for Nonlinear Programming," Report SOL 86-1, Department of Operations Research, Stanford University, 1998.

9. R. Fletcher, *Practical Methods of Optimization, Second Edition*, Wiley, Chichester, UK, 1987, pp. 55-57.

10. C. M. Traweek, "Optimal Spatial Filtering for Design of a Conformal Velocity Sonar Array," Ph.D. Dissertation, The Pennsylvania State University, May 2003.

APPENDIX
FLOWCHART OF THE ARRAY SHADING SQP ALGORITHM

This appendix presents a flowchart of the AS-SQP algorithm. The variables in the flowchart are defined as follows:

N :   Number of unknowns (hydrophones)
M :   Number of constraints (one equality constraint)
W(N) :   Hydrophone weight vector
X(N) :   Search direction vector
OBJ: Objective function
G(N) :   Objective function gradient vector
H(N,N) :   Hessian matrix of the objective function
A(M,N) :   Constraint matrix (each element = 1)
B(M) :   Vector containing RHS of constraints (= 1)
BL(N) :   Vector containing lower bound of search direction vector
BU(N) :   Vector containing upper bound of search direction vector
IPRINT :   Output indicator for subroutine DIQP
MAXITR :   Maximum number of iterations permitted in subroutine DIQP (= 3*N)
IEQ :   Number of equality constraints (= 1)
MAXMAJ :   Maximum number of major iterations permitted for optimization
STEPU :   User-defined step size
STEPI :   Variable containing the maximum step size allowed by weight vector at current iterate
WDIFF :   Difference between current and previous iterates of weight vector (sum of absolute values of components of difference vector)
IMAJ :   Major iteration counter

The program starts by reading the number of unknowns and constraints. The lower- and upper-bound constraints for the components of the search direction vector are initially set to -1 and 1, respectively, in accordance with condition (20c). After the constraint variables and program parameters are set, the weight vector and search direction vector are initialized. The optimization commences with the evaluation of the objective function, gradient, and Hessian at the initial point $\mathbf{w}^{(0)}$. These quantities, derived in section 2, are implemented in subroutine FNOBJ. After the objective function, gradient, and Hessian are evaluated, the QP subproblem (20) is carried out through use of subroutine DIQP, developed by Bunch and Kaufman (reference 7).

After evaluation of the QP subproblem, the step size is determined in accordance with formula (33). Upon determination of the step size, the weight vector is updated via formula (26). For each component of the weight vector that is active (i.e., equal to zero), the lower bound for the corresponding component of the search direction vector is set to zero. For the remaining components, the lower bound is set to -1. The major iteration counter IMAJ is then updated, and the sum of the absolute values of the differences of the weight vector components at the current and previous iterates is evaluated and stored in the variable WDIFF; i.e.,

$$\text{WDIFF} = \sum_{m=1}^{M} \left| w_m^{(k+1)} - w_m^{(k)} \right|. \tag{A-1}$$

If WDIFF is less than a preset tolerance for two successive iterations, an optimum solution is attained and the program terminates. In addition, if the maximum number of iterations is achieved before an optimum solution is obtained (i.e., IMAJ = MAXMAJ), the program terminates. The above procedure is repeated until either a convergent result is obtained or the maximum allowed number of iterations is reached. The computer program is written in FORTRAN 95 and is run in double precision.

```
                    ┌─────────────┐
                    │  Start SQP  │
                    └──────┬──────┘
                           │
                           ▼
                     ╱─────────────╲
                    ╱  Read M, N    ╲
                    ╲               ╱
                     ╲─────────────╱
                           │
                           ▼
                  ┌──────────────────┐
                  │ SET CONSTRAINT   │
                  │ VARIABLES        │
                  └────────┬─────────┘
                           │
                           ▼
                  ┌──────────────────┐
                  │ SET              │
                  │ PARAMETER        │
                  │ S               │
                  │ STEPU            │
                  │ IPRINT           │
                  └────────┬─────────┘
                           │
                           ▼
                ┌──────────────────────┐
                │ INITIALIZE WEIGHTS AND│
                │ SEARCH DIRECTION      │
                └──────────┬───────────┘
                           │
                           ▼
                   ┌───────────────┐
                   │  IMAJ ← 0     │
                   └───────┬───────┘
                           │
        ┌───┐              │
        │ 2 ├──────────────┤
        └───┘              ▼
                   ┌───────────────┐
                   │  Call FNOBJ   │
                   └───────┬───────┘
                           │
                           ▼
                   ┌───────────────┐
                   │  Call DIQP    │
                   └───────┬───────┘
                           │
                           ▼
                  ┌─────────────────┐
                  │ STEPI ← STEPU   │
                  └────────┬────────┘
                           │
                           ▼
              ┌─────────┬──────────┐
              │ I ← 1   │          │   yes      ┌───┐
              ├─────────┤ I > N?   ├───────────▶│ 1 │
              │ I ← I+1 │          │            └───┘
              └─────────┴────┬─────┘
                             │ no
                             ▼
                        ╱─────────╲
                  no   ╱  X(I) > 0? ╲   yes
              ┌───────╲            ╱──────────┐
              │        ╲─────────╱            │
              ▼                               │
   ┌──────────────────────┐                  │
   │ STEMP ← −W(I)/X(I)   │                  │
   └──────────┬───────────┘                  │
              │                               │
              ▼                               │
         ╱──────────╲                         │
   no   ╱ STEMP <     ╲   yes                 │
  ┌────╲   STEPI?     ╱────────┐              │
  │     ╲───────────╱          │              │
  │                            ▼              │
  │               ┌──────────────────┐        │
  │               │ STEPI ← STEMP    │        │
  │               └────────┬─────────┘        │
  │                        │                  │
  └────────────────────────┴──────────────────┘
```

①

WDIFF ← 0

| I ← 1 | I > N? |
| I ← I+1 | |

yes →

no

WTEMP ← W(I)
W(I) ← W(I) + STEPI*X(I)
WDIFF ← WDIFF + ABS(W(I) −WTEMP)

W(I) = 0?

no → BL(I) ← −1

yes → BL(I) ← 0

IMAJ ← IMAJ+1

WDIFF < 1.E-12?

no

yes → Print Results → Stop

IMAJ = MAXMAJ?

no → ②

yes → Print Results → Stop

INITIAL DISTRIBUTION LIST

Addressee	No. of Copies
Office of Naval Research (D. Todoroff, J. McEachern, C. Traweek (10))	12
Defense Technical Information Center	2